Kathleen Fiegert

Konzeption und prototypische Umsetzung einer webbasierenden Anwendung für die Visualisierung von Maschinenzuständen und Fertigungsprozessdaten an flexiblen Montageanlagen.

eingereicht als

BACHELORARBEIT

an der

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

Mathe Physik Informatik

Chemnitz, 23.09.2009

Erstprüfer: Betreuer FH: Prof. M. Geißler
Zweitprüfer: H. Hartleib

Vorgelegte Arbeit wurde verteidigt am: 23.09.2009

Bibliographische Beschreibung

Fiegert, Kathleen:
Konzeption und prototypische Umsetzung einer webbasierenden Anwendung für die Visualisierung von Maschinenzuständen und Fertigungsprozessdaten an flexiblen Montageanlagen. - 2009. - 31 S.
Mittweida, Hochschule Mittweida (FH), Fachbereich Mathe Physik Informatik, Bachelorarbeit, 2009

Kurzreferat

Diese Bachelorarbeit befasst sich mit der Untersuchung einer geeigneten Webanwendung für die Darstellung von Maschinenzuständen und Fertigungsprozessdaten. Zu Beginn erfolgt eine Untersuchung zu den gängigsten Webframeworks, die mit der Programmiersprache Java Webanwendungen erzeugen können. Dabei liegen die Schwerpunkte auf Ajaxfunktionalität, unter Verwendung des Model View Controller's (MVC) und Unterstützung der Internationalisierung (I18N). Nach einer ersten Vorauswahl werden besonders gut geeignete Webframeworks genauer untersucht. Mit dem ausgewählten Framework erfolgt eine schrittweise Erstellung und Umsetzung eines Prototyps. Dazu gehört die Erstellung einer Datenbank, in der die angezeigten Maschinenzustände gespeichert werden. Für die Verwendung ohne SPS ist die Erstellung entsprechender Testdaten mit Zufallsgeneratoren für den Prototyp erforderlich. Neben einem Funktionstest erfolgt ein Leistungstest, der mit entsprechenden Tools durchgeführt wird. Während der gesamten Programmentwicklung ist stets darauf zu achten, dass die Verwendung an flexiblen Montageanlagen erfolgen soll. Des Weiteren ist zu bedenken, dass die Programmierung der Webanwendung mit wenigen Quellcodeänderungen für verschiedenartige Fertigungsanlagen wieder verwendbar ist. Bei der Visualisierung wird speziell für die Touchscreendisplays im Industrieeinsatz auf eine ergonomische GUI-Gestaltung Wert gelegt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DB	Datenbank
GC	Graphical Context
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JAAS	Java Authentication and Authorization Service
JavaEE	Java Enterprise Edition
JMS	JavaMessageService
JSNI	JavaScript Interface
JVM	Java Virtual Machine
MVC	Model-View-Controller
OS	Operating System
RAP	Rich Ajax Platform
RIA	Rich Internet Application
RPC	Remote-Procedure-Calls
RWT	RAP Widget Toolkit
SQL	Structured Query Language
SWT	Standard Widget Toolkit
UI	User Interface
URL	Uniform Resource Locator
WAR	Web Application Archive
XML	Extensible Markup Language

0 Einleitung

In der Bachelorarbeit soll ein Prototyp einer Webapplikation für die Visualisierung von Maschinenzuständen und Fertigungsprozessdaten an flexiblen Montageanlagen erstellt werden. Dies wird in der Firma SITEC Industrietechnologie GmbH, Abteilung Prozesssteuerung, mit der Programmiersprache Java erarbeitet. Die Webanwendung wird dem Schichtleiter die Möglichkeit geben, sich von einem Rechner im Netzwerk der Firma auf eine Montageanlage zuzuschalten. Dieser Zugriff erfolgt über einen Browser. Dieses Programm soll mit geringem Installationsaufwand und wenig technischem Fachwissen auf dem Rechner des Schichtleiters konfiguriert werden können. Außerdem ist das Programm so flexibel, dass es ohne großartige Änderungen am Quelltext für weitere Anlagen anpassbar ist. Das ist besonders für die Verwendung im Bereich des Sondermaschinenbaus wichtig, in dem die Firma SITEC Industrietechnologie GmbH stark vertreten ist.

In Kapitel 1 werden für die Visualisierungsmöglichkeiten verschiedene webbasierende Lösungen untersucht und anschließend eine Webanwendung nach ausgewählten Kriterien bezüglich Entwicklungssprache, Leistung und Architektur ausgewertet. Mit der Entscheidung für ein entsprechendes Framework wird in Kapitel 2 eine Anwendung entworfen. Die Visualisierung der Anwendung ist für ein Touchpanel zu konzipieren, dabei wird eine ergonomische GUI-Gestaltung berücksichtigt. Danach erfolgt die prototypisch Umsetzung im 3. Kapitel. Zudem ist die Erzeugung von speziellen Testdaten vorgesehen, welche in einer dafür angelegten Datenbank (DB) archiviert werden sollen. Abschließend werden einige Tests in Bezug auf Fehlersicherheit und Laufzeitverhalten in Kapitel 4 durchgeführt und ausgewertet. Eine Einschätzung, wie die gesamte Arbeit verlaufen ist sowie Funktionen, die für die Zukunft möglich wären, sind in der Zusammenfassung dargestellt.

1 Analyse der Verfahren

Ein detaillierter Vergleich aller Verfahren im Rahmen dieser Arbeit ist nicht möglich, da es sehr viele Verfahren gibt, webbasierende Anwendungen zu gestalten, und sie auch zunehmend unüberschaubarer werden. Aus diesem Grund beschränkt sich der Autor auf einige Aspekte weit verbreiteter aktueller Frameworks. Es soll hier ein erster Überblick gegeben werden, um eine Vorauswahl zu ermöglichen.

Bei der Auswahl eines geeigneten Frameworks lag der Schwerpunkt auf der Entwicklungssprache Java und der Programmierer sollte keine Kenntnisse in anderen Sprachen benötigen. Java ist portierbar, also hardware- und betriebssystemunabhängig, robust, sicher, hat viele Standardfeatures und ist dynamisch und modular, im Gegensatz zu anderen Programmiersprachen. Ein weiterer Vorteil von Java ist der browserbasierte Ansatz, wo JavaScript als Laufzeitumgebung in den Clients zum Einsatz kommt. Außerdem ist Java statisch typisiert, so können mehr Fehler zur Zeit der Kompilierung abgefangen werden und die Refaktorierung ist einfach.

Für die Entwicklung großer Webanwendungen ist es von Nutzen, Webframeworks zu verwenden. Diese verallgemeinern die Plattform JavaEE und verbessern die Übersichtlichkeit für den Entwickler. Des Weiteren geben sie eine grundlegende Struktur der Webanwendung vor, zu welcher es sonst keine Vorgaben gibt. Dem Entwickler kann durch die Software-Pakete der Webframeworks für die Internationalisierung, für Datenbankzugriffe und für die Validierung viel Arbeit abgenommen werden.

In Bezug auf die Lizenzierung werden nur OpenSource Lösungen in Betracht gezogen. Außerdem muss das Framework „Asynchronous JavaScript and XML“ (Ajax) Funktionalitäten besitzen, welche in den meisten aktuellen Anwendungen bereits integriert sind. Der Vorteil von Ajax ist die asynchrone Datenübertragung zwischen Server und Client. Dabei wird auf einer HTML-Seite eine HTTP-Anfrage durchgeführt, ohne dass die Seite neu geladen werden muss. Überdies ist Ajax unabhängig von Betriebssystemen und Webbrowsern.

Großer Wert wird auch auf den Model-View-Controller gelegt, welcher eine Vorlage für die Architekturstrukturierung bei der Entwicklung von Software ist. Mit dem MVC wird die Präsentationsschicht in Datenmodell, Präsentation und Programmsteuerung dreigeteilt. Die Geschäftslogik, welche sich in der Geschäftsschicht befindet, wirkt als Schnittstelle zwischen der Präsentation und der Datenbank. Die Realisierung der Präsentationsschicht erfolgt durch den Browser des Clients. Die Programmsteuerung nimmt also die Benutzer-Requests entgegen und stößt anschließend das Datenmodell an. Danach benachrichtigt die Programmsteuerung die Präsentationsansicht, welche die entsprechenden Daten bezieht und anzeigt. Die Abbildung 1-1 verdeutlicht dies. Die Komplexität der Präsentationsschicht kann durch das MVC erheblich reduziert werden, da Präsentationsänderungen vom Datenaustausch der Programmsteuerung und des Datenmodels gesondert verarbeitet werden. Es kommt also zu einer sauberen Trennung der Anwendung und Darstellung, wodurch eine einfache Wartung und Erweiterung der Applikation möglich ist. Außerdem ergibt sich dadurch die Möglichkeit, einzelne Komponenten wieder zu verwenden. Die Prozesse Präsentationsänderungen und Datenaustausch erfolgen gemeinsam.

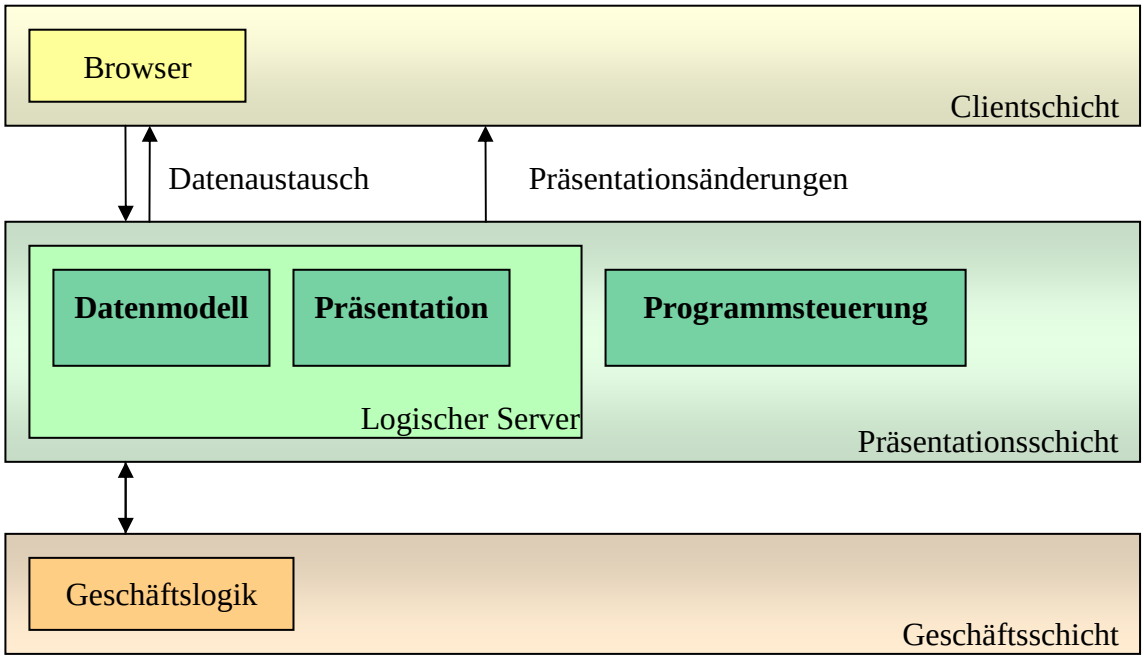


Abbildung 1-1: Die Drei-Schicht-Architektur in Kombination mit dem MVC[1]

[1] Quelle: Eigene Darstellung

Bei der Erstellung von Softwareanwendungen ist es zweckmäßig, die Internationalisierung (I18N) zu beachten. Ein Programm ist am besten so zu gestalten, dass es ohne Änderungen im Quellcode an andere Sprachen angepasst werden kann. Dabei werden Textdaten, Datum, Währungssymbole und andere spezielle Einstellungen nicht im Quellcode fest codiert, sondern in externen Dateien abgelegt und durch Variablen zur Laufzeit dynamisch geladen.

Für die Applikation wurden verschiedene Testmöglichkeiten der einzelnen Frameworks ermittelt. Da alle ausgewählten Frameworks in Java programmiert werden, steht zum Testen des serverseitigen Codes das Framework JUnit bei allen Applikationen zur Verfügung. Neben den Unit Tests gibt es noch funktionale Tests, welche die Anwendungen auf der Clientseite überprüfen. Damit kann auch das JavaScript auf Fehler untersucht werden.

Weiterhin ist von Interesse, ob der Logikschwerpunkt auf der Server- oder auf der Clientseite liegt. Bei Anwendungen, wo der Schwerpunkt auf der Serverseite liegt, ist der Nachteil, dass bei jeder Änderung eine Anfrage vom Client zum Server erfolgt. Dies ist bei clientseitiger Last nicht der Fall. Jedoch müssen hier dafür auch auf dem Client Installationen vorgenommen werden, welche bei der serverseitigen Logik nicht notwendig sind.

Besonders zu beachten sind die Rich Internet Applications (RIA), auch als Rich Web Application bekannt. Eine Web-Applikationen muss nicht immer eine RIA sein. Es handelt sich nicht um schlichtes HTML, sondern es geht neben dem Aussehen auch um die Bedienbarkeit der Anwendung. RIAs haben einen direkten Zugriff auf jeden Teil der Darstellung, eine unmittelbare Verarbeitung von Benutzereingaben und eine asynchrone Kommunikation mit dem Server in Echtzeit. Beispielsweise erfahren Benutzer bei HTML erst ob die Aktion erfolgreich verlaufen ist, wenn die neue Seite fertig geladen ist. Anders ist dies bei RIAs, wo die Rückmeldungen auf Aktionen unmittelbar verfügbar sind. Ein weiteres Beispiel ist, dass beim Überprüfen von Formulareingabefeldern mittels HTML oft ein Roundtrip zum Server benötigt wird und bei HTML werden Fehlermeldungen oftmals nicht dort wo sie auftreten, sondern erst auf globaler Ebene angezeigt.

Die Tabelle Webframeworks, im Anhang 1, zeigt die Gegenüberstellung ausgewählter Frameworks. Google Web Toolkit (GWT) und Rich Ajax Platform (RAP) werden wegen ihrer RIA Funktionalität, der großen Anzahl von Standardfeatures und der großen Nutzeranzahl im nächsten Abschnitt genauer untersucht.

1.1 Vergleich von GWT und RAP

Das RAP- und das GWT-Framework sind beide Rich-browserbasierend, sie können beide mit Eclipse arbeiten und nutzen Ajax. Außerdem unterstützen beide die Entwicklungssprache Java und es werden keine JavaScript-Kenntnisse vorausgesetzt. Die Frameworks basieren auf gängigen Programmiermodellen, die sich bei grafischen Benutzeroberflächen deutlich bewährt haben. Die RAP nimmt klassischen SWT-Code und generiert damit automatisch Ajax-Benutzeroberflächen, somit sind SWT-Kenntnisse sehr nützlich. Ähnlich und doch anders arbeitet das GWT. Hier wird in Swing programmiert, welches schöne UI-Komponenten für das Web bietet. Im weiteren Verlauf werden die beiden Konzepte analysiert und aufgezeigt, welche Stärken und Schwächen die jeweiligen Ansätze mit sich bringen.

1.2 Compilierung und Fehlerbehebung

Der gehostete GWT Modus führt Javacode aus und stellt die Widgets in einem Host-Fenster dar. Der GWT- und RAP-Compiler transformiert die Applikation in JavaScript, und führt diese in einem Webbrowser aus. Diese Übersetzung wird in Abbildung 1-2 gezeigt. Diese kompilierte Anwendung nennt man den GWT Webmodus. Leider können in GWT nicht alle Java-Bibliothek-Funktionalitäten in JavaScript übersetzt werden. Beispielsweise können keine Klassen nachträglich dynamisch geladen werden und auch die automatische Speicherbereinigung ist nicht möglich. Da viel mehr der Entwicklungszeit beim Ausführen im gehosteten Modus benötigt würde, geschieht die Interaktion in der GWT-Applikation, ohne diese in JavaScript zu übersetzen. Jederzeit kann die Applikation von einer Java integrierten Entwicklungsumgebung (IDE), wie Eclipse, editiert, ausgeführt und ausgetestet werden. In der IDE ist es möglich, den clientseitigen GWT-Code und auch den serverseitigen Javacode auszutesten. Beim Ausführen einer Applikation im gehosteten Modus führt die Java Virtual Machine

(JVM) den Applikationscode aus und benutzt die GWT-Installation, um ein eingebettetes Browserfenster zu automatisieren. Der Server startet wie der GWT-Client im gehosteten Modus. In RAP gibt es jeweils eine Startkonfiguration für Server und Client, also einen internen Browser und einen externen Browser. Es können auch unterschiedliche Browser verwendet werden oder ein Browser, der auf einer anderen Maschine läuft und auf dieselbe URL zeigt.

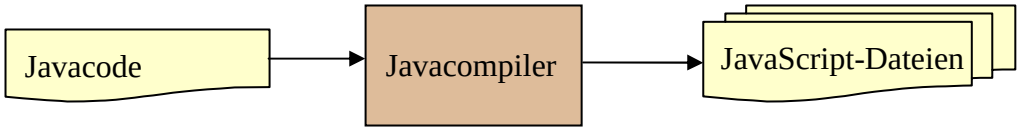


Abbildung 1-2: Umwandlung von Javacode in JavaScript durch den Compiler[2]

1.3 Erstellen des Nutzerinterfaces

Unterschiede bestehen in den UI Komponenten, bei RAP haben die meisten SWT-Widgets RAP Widget Toolkit (RWT) Gegenstücke, außer StyledText, FormText und Graphical Context (GC). Vieles von JFace ist bereits implementiert, wie beispielsweise Tabellen, Dialoge, Farben, Schriftarten, Inhalt- und Label Providers. Drag und Drop wird zu Zeit noch geplant und es besteht die Möglichkeit zur Erstellung eigener Widgets. Weiterhin unterstützt RAP CSS-Dateien (Cascading Style Sheets) und durch die Qooxdoo-JavaScript-Bibliothek, die meist verwendeten Browser wie: Firefox, Microsoft Internet Explorer, Opera und viele weitere. Die GWT-Komponenten unterstützen neben verschieden Arten von Buttons, Hyperlinks, Tabellen und Listen auch die Erstellung eigener Widgets. Zum Ausführen des JavaScripts wird hierbei das JavaScript Interface (JSNI) benutzt. Die GWT-Gestaltung basiert auf den Standard CSS, wo der eigentliche Inhalt von der optischen Gestaltung getrennt wird. Des Weiteren ist es in CSS möglich, unterschiedliche Darstellungen für verschiedene Ausgabemedien, beispielsweise bei Projektionen oder bei unterschiedlichen Sprachen, anzugeben und Elemente frei zu positionieren. Wie auch bei den RAP-UI-Komponenten werden die aktuellen meist verbreiteten Browser unterstützt. GWT unterstützt darüber hinaus Funktionalitäten des Zurück-Buttons durch den Browserverlauf, wobei die AJAX-Anwendungen für den Zurück-Knopf des Browsers keine Unterbrechung des

[2] Quelle: Eigene Darstellung

Programms brauchen. In der RAP gibt es keine Unterstützung für die Zurückstellung des Browsers, Qooxdoo bietet zwar diese Funktionalität an, aber RAP nutzt sie nicht.

GWT-Widgets basiert auf den CSS für die visuelle Gestaltung und auch RAP wird von den CSS Dateien unterstützt. Standard CSS dient zur Erstellung einheitlicher und globaler Anwendungen und für das Erstellen eigener Widgets. Es bietet auch die Möglichkeit komplexe Gestaltungen, beispielsweise mit „MenuBar“ für die Erstellung von Menüs, vorzunehmen.

Mit GWT können viele kleine Images zu einem einzelnen großen Image für einen effizienten Transport zum Browser kombiniert werden. Die sogenannten ImageBundles machen das Benutzen von Images effizienter, weil es schneller geht, eine große Datei als viele kleine zu laden. In der RAP sind ImageBundles noch nicht verfügbar, aber jedoch bereits geplant.

1.4 Kommunikation mit einem Server

GWT unterstützt mehrere Wege zur Kommunikation mit einem Server mittels HTTP. GWT-RPC-Framework kann genutzt werden, um Aufrufe zu Java-Servlets transparent zu machen und GWT auf Objektserialisierung achten zu lassen. Beim Zugriff vom Browser auf die Methoden werden automatisch die Argumente von GWT-RPC serialisiert. Danach ruft er die dazugehörige Methode auf dem Server auf und dann wird der Rückgabewert für den Clientcode deserialisiert. Außerdem kann das GWT-RPC polymorphe Klassen-Hierarchien und Objektdarstellungen periodisch bearbeiten und auch Ausnahmen während der Verbindung abfangen. Alternativ kann das HTTP von GWT Client-Klassen zum Bauen und Senden benutzerdefinierter HTTP-Anfragen genutzt werden.

GWT kann schneller sortieren als RAP, weil mehr Informationen auf der Clientseite gecached werden und mehr Verarbeitung auf dem Client stattfindet. Aber RAP ist etwas effizienter beim Laden der Verzeichnisse und beim Starten der Applikation. RAP übermittelt nur die Informationen, die zum Darstellen der sichtbaren Spalten in der Tabelle nötig sind. In der GWT-Applikation wird nur dann der ganze Tabelleninhalt

geladen, wenn er vollständig benötigt wird und auf dem Client gecached. Ansonsten wird nur das geladen was unbedingt gebraucht wird, um somit die Aktualisierungszeit zu reduzieren. GWT ist deswegen genügsamer und potenziell schneller. Da RAP die Aufgaben von dem Client auf dem Server ausführt, werden mehr und häufiger Informationen mit dem Server ausgetauscht. Die Abbildung 1-3 und Abbildung 1-4 verdeutlichen dies.

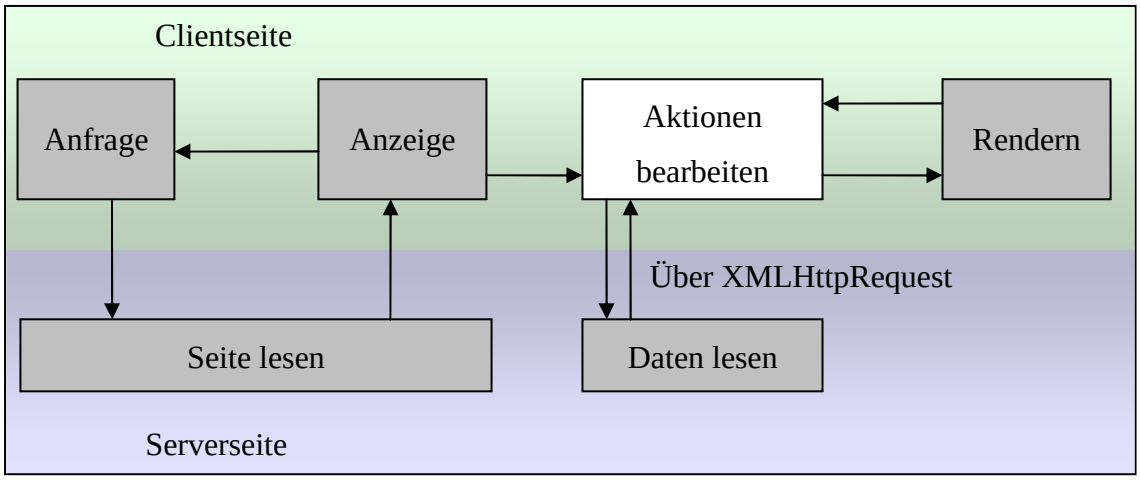


Abbildung 1-3: GWT- Kommunikation[3]

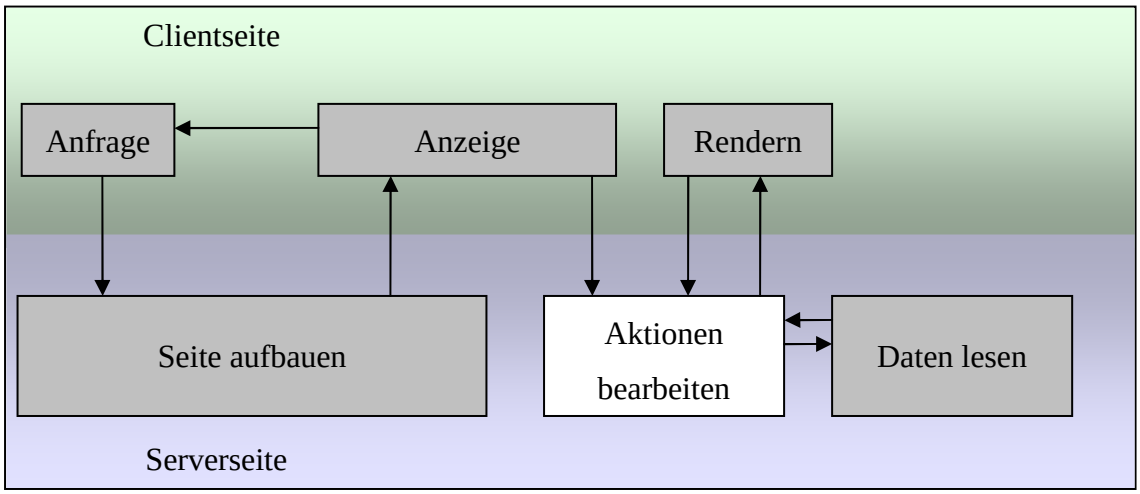


Abbildung 1-4: RAP – Kommunikation[4]

[3] Quelle: Eigene Darstellung
[4] Quelle: Eigene Darstellung

Der Schwerpunkt der Logik liegt bei GWT auf der Clientseite. Deswegen müssen für die serverseitige Logik, Interfaces und Klassen für Remote-Procedure-Calls (RPC) erstellt werden. RAP ist in dieser Hinsicht einfacher zu entwickeln als GWT. GWT ist dafür flexibler, da die Informationen optional auf dem Client gecached werden und Informationen aus dem Hintergrund hervorgerufen werden können.

1.5 Architekturperspektiven

Die GWT-RPC-Grundlage ist eine Methode zur Kommunikation von Client zum Server. RPC wird zum Lösen von Aufgabenteilen der Anwendung genutzt. Architektonisch kann RPC auf zwei Wegen genutzt werden. Zu unterscheiden sind hierbei nur die Aufgaben und die Architektur der Anwendung. Der clientseitige Code wird "front end" und der auf dem Server "back end" ausgeführt. Die Serviceimplementierung hat mehr allgemeingültige APIs, welche lose an eine spezifische Applikation gekoppelt sind. Die Servicedefinitionen haben einen direkten Datenbankzugriff durch JDBC, Hibernate oder auch Dateien im Server-Datei-System. Bei GWT kann der Aufbau der Anwendung durch die Reduzierung der Schichtenanzahl sehr effizient gestaltet werden. Ein typischer Aufbau solch einer Applikation ist in der Abbildung 1-5 dargestellt.

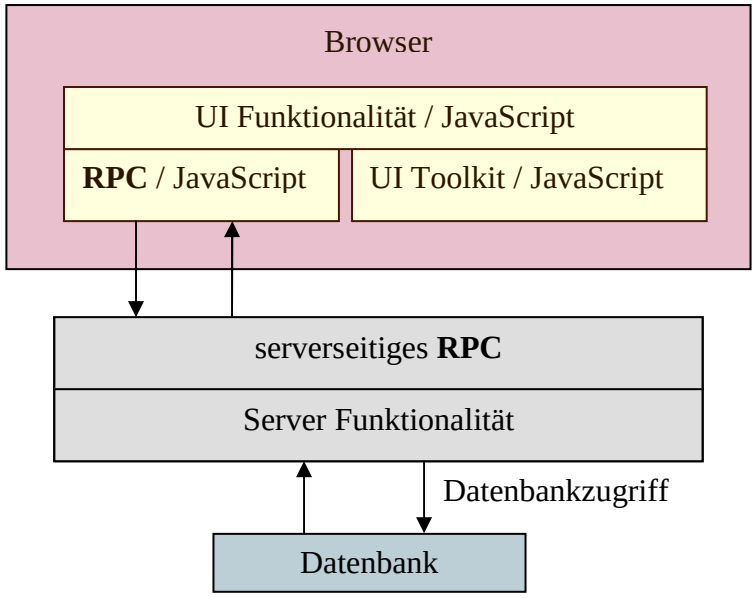


Abbildung 1-5: Remote-Procedure-Calls[5]

[5] Quelle: Eigene Darstellung

Der mit GWT eingebettete Webserver im gehosteten Modus ist nur für das Debugging der Webapplikation gedacht. Sobald die Entwicklung der Webapplikation fertig ist, kann sie auf dem Server ausgeführt werden. Wenn die Webapplikation nur aus statischem Inhalt besteht, hat der Webserver viel zu tun, deshalb benutzen die meisten GWT Webapplikationen RPC und Java Servlets. Für diese Art der Applikationen muss ein Servlet Container zum Ausführen im „back end“ gewählt werden. GWT hat bei der Entwicklung keinen eigenen Servlet-Container, es werden aber viele Produkte als Servlet Container angeboten, wie beispielsweise Apache Tomcat.

Bei GWT gibt es noch eine andere Methode für die Kommunikation zwischen Server und Client, diese erlaubt, durch die Nutzung eines Servlets, das Erhalten von XML-Daten mit HTTP vom Server. Dann kann dieses XML mittels GWT-XML-Parser auf der Clientseite ausgewertet werden. Diese Methode ermöglicht die Unabhängigkeit von der Serverseite.

GWT braucht für den Entwurf der Architektur von der Präsentationsschicht sehr lange. Beim Entwickeln großer und komplexer Webanwendungen mit GWT entstehen viele Klassen. In GWT gibt es keine weiterentwickelten Widgets. Hier besteht eine server-seitige GWT Architekturabhängigkeit.

Das Javascript muss die Objekte erst serialisieren, bevor sie vom Client zum Server gesendet werden können, weil es mit dem XMLHttpRequest sonst nicht funktioniert, da nur eindimensionale Parameter unterstützt werden. Die serialisierten Objekte werden gebraucht, um die Zeichnungsberechnungsleistung der Clients zu realisieren. Bei großen Objekten kann die Performance nicht garantiert werden, es würde also zu langen Wartezeiten kommen.

Bei der RAP handelt es sich um ein Frontend Framework, somit ist es näher am Benutzer als am System. Die RAP läuft auf dem Server und dem Client, und ist mit dem Standardbrowser zugänglich. Dabei wird der Javacode auf dem Server ausgeführt und die statische JavaScript-Bibliothek auf dem Client genutzt. Die Abbildung 1-6 zeigt eine mögliche Präsentation der Komponenten, die an der RAP-Stapelbildung beteilig sind.

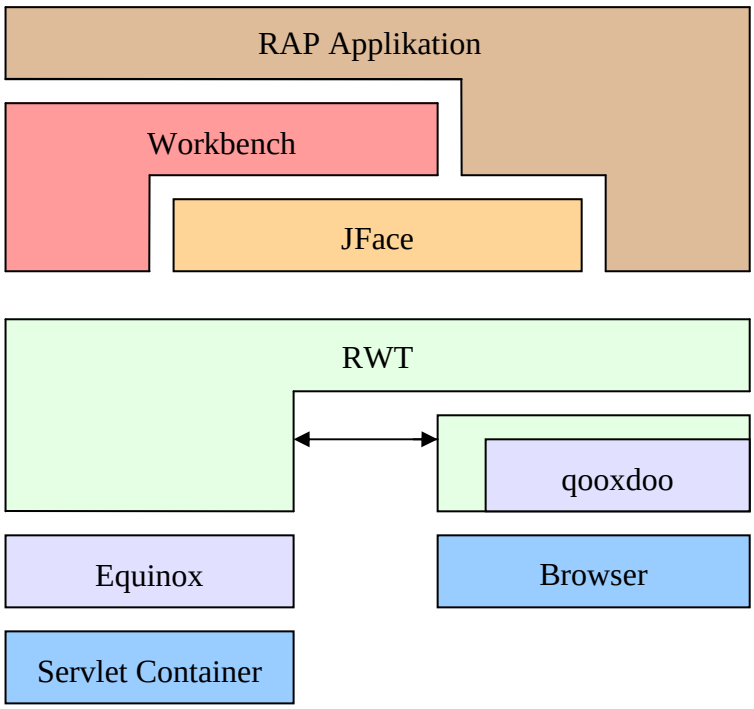


Abbildung 1-6: PAP-Architektur[6]

Die Workbench dient der Unterstützung der Kommunikation von Aktivitäten, Kommandos und dem Inhalt. Die Benutzerschnittstellenkomponenten sind hierarchisch und logisch gruppiert, wie beispielsweise das Managen einer Seitenansicht. JFace ist eine Java Application Framework, das eine Ergänzung des Standard Widget Toolkits darstellt, mit der Aufgabe, Klassen und Frameworks anzufügen. Außerdem ist die RWT-Bibliothek vertreten, wobei jedes RWT Widget einen serverseitigen und einen clientseitigen Teil besitzt. Die Kommunikation geschieht über den Ajax Request, dabei werden die Widget-Hierarchien auf dem Client sowie dem Server synchronisiert.

Für RAP gibt es zwei existierende Laufzeitumgebungen. Die Eine befindet sich auf der Serverseite und versorgt den Haupt-RAP-Stack, die Andere bleibt auf der Clientseite und versorgt die Benutzerschnittstellen. Auf der Serverseite der RAP gibt es auch eine Java Virtuell Machine, die auf zwei Plattformen läuft, dabei muss das Operation System sich nicht um das Rendern der Widgets kümmern.

Die eine Plattform ist Equinox, eine OSGI-Implementierung von Eclipse. Equinox stellt die Laufzeitumgebung für die Eclipse-Bundle bereit. RAP implementiert diese Laufzeit-

[6] Quelle: Anlehnend an: http://www.eclipse.org/rap/about.php.

umgebung zur Konstruktion von Ajax-fähigen Webanwendungen in Java, welche dann vom Server eingesetzt werden kann.

Bei der anderen Plattform handelt es sich um einen Servlet-Container, welcher beispielsweise für die Anfragenbearbeitung verantwortlich ist. Ein solcher Servlet-Container ist der Tomcat-Server, welcher hier nur für die Kommunikation mit dem Webclientbrowser benutzt wird. Die Anfrage wird durch eine Servlet, welches als Bridge fungiert, aufgelesen und danach von der Webanwendung an den Server innerhalb des Equinox-Systems weitergereicht. RAP-Anwendungen können mittels „Standard Web Archivs“ (WAR), mit einem eingebetteten Equinox, in die Webanwendung eingesetzt werden.

Der Browser läuft auf der Clientseite, welche sich auf dem Client-OS befindet. Der Browser arbeitet wie eine JVM, der einen eigenen Standardservice der Oberschicht unterstützt. Qooxdoo dient als Benutzerschnittstellenteil auf der Clientseite, es wird als Teil vom RAP-Bundle genutzt. Dabei erstellt dieses Open Source JavaScript Framework die benötigten JavaScript Kommandos für das Rendern der geeigneten Widgets und diese Kommandos senden dann mittels Ajax zum Browser.

Die RAP-Plug-in-Entwicklung auf dem Server wird durch OSGi Bundle mit integrierten HTTP-Service realisiert. Bei der Entwicklung mit J2EE Anwendungsservern wird die Servlet Bridge genutzt, leider ist bei RAP keine Entwicklung eines reinen Webserver möglich. Bei der GWT-Plug-in-Entwicklung kann der Server zu einem OSGi basierenden Server entwickelt werden, welcher Plug-ins benutzt. GWT kann zu einem J2EE Anwendungsserver als WAR-Datei entwickelt werden. Die Entwicklung einer einfachen Applikation zu einem einfachen Webserver ist möglich. Die GWT-J2EE-Serverentwicklung ist ähnlich der GWT-Plug-in-Entwicklung. Es gibt nur Unterschiede im J2EE basierten Aufbau und beim Entwicklungsprozess. So wird hier der Aufruf Javac zum Kompilieren des serverseitigen Java-Codes aufgerufen und das Entwerfen der WAR-Datei beinhaltet clientseitig JavaScript und serverseitig Java.

Bei dem Plug-in-basierten Aufbau- und Entwicklungsprozess werden bei RAP alle Plug-ins in ein Feature integriert. Bei GWT wird die Bibliothek in eine JAR-Datei geschoben, der GWT-Compiler für die Compilierung von Java zu Javascript aufgerufen

und dann in ein Plug-in geschoben. So ist der PDE-Bauprozess für RAP und GWT einfacher. Um den PDE-Bau zu starten, müssen Bundles generiert werden, dabei wird neu formatierter Code in das PDE-Format gebracht, anschließend wird der PDE-Bau aufgerufen und das PDE-Bauergebnis entpackt und neu formatiert. Danach wird das Bauergebnis im Anwendungsserver aktualisiert, somit kann das Entwicklungsscript auf dem Anwendungsserver ausgeführt werden.

Mit RAP lässt sich also ein OSGi-basierter Server leichter entwickeln als mit GWT. Mit GWT kann dafür leichter einen Anwendungsserver entwickelt werden als mit RAP. Die Applikationen können lokal und auf dem Server entwickelt werden.

1.6 Testen

Da RAP- und GWT-Anwendungen auf Java basieren kann zum Testen des server-seitigen Codes JUnit benutzt werden. Wie in jeder Java-Anwendung ist auch das Testen der clientseitigen Anwendung durch die Unterstützung eines API (Application Programming Interface), die als eine Bridge zwischen JUnit und GWT (oder RAP) Umgebung agiert, möglich. Dafür beinhaltet die Clientseite von RAP-Anwendungen eine spezielle „RAPTestCase“ Klasse, welche die JUnit-Integration unterstützt. Die GWT-Anwendungen beinhalten spezielle „GWTTestCase“ Klassen, die auch die JUnit-Integration der Clientseite unterstützen, somit kann im Debugger und im Browser getestet werden, aber dafür sind spezielle GWT-Module notwendig. Weiterhin bietet GWT eine optionale Unterstützung um asynchrone RPC-Tests zu absolvieren und für die Spezifikation der maximalen Durchführungsdauer. GWT unterstützt kein Multithreading oder Blockung.

Mit speziellen Tests ist es möglich, Fehler in der Anwendung zu lokalisieren. Die Korrektur muss manuell durchgeführt werden. Bei der Fehlerdiagnose wird unter Verwendung der Java-Bibliothek HTML-Unit ein Browser simuliert. Außerdem besteht die Möglichkeit, einen Browser, wie Internetexplorer oder Firefox, anzusteuern. Mit dem Tool Webtest können beispielsweise Fehler im Javacode gezielt gesucht und gefunden werden sowie das Finden von JavaScript-Fehlern ist mit diesem möglich. GWT wird zu 100 % von Webtest unterstützt und für qooxdoo ist es schon zu großen

Teilen nutzbar. Webtest kann feststellen, ob asynchroner JavaScript-Code noch ausgeführt werden muss und gegebenenfalls darauf warten. Damit wird der Aufwand im Testcode reduziert. Die größten Stärken von Webtest liegen in der einfachen Anwendung ohne externe Abhängigkeiten. Dies bietet die Möglichkeit, schnelle und robuste Tests zu schreiben.

1.7 Sicherheit

Jede Art von Sicherheits-Framework ist hier verwendbar, die dabei auftretenden Probleme müssen vom Entwickler behoben werden. RPC, welches durch GWT definiert ist, findet auf der Serverseite im Java-Servlet Anwendung. Das bedeutet, die GWT-Anwendungen werden dann auf der Oberfläche von einem JavaEE-Anwendungsserver ausgeführt. In diesem Fall könnte zur Unterstützung der Authentifizierung der Java Authentication and Authorization Service (JAAS) benutzt werden. Der Authentifizierungsmechanismus wird so durch die Java-Servlet-Spezifikation unterstützt. Es müssen nur all die HTML-, JavaScript- und CSS-Dateien für die Anwendung in einer WAR-Datei platziert werden. Außerdem wird eine Unterstützung einer Login-Seite in der WAR-Datei gebraucht. Auf diesem Weg kann man den Namen von entfernten Nutzern in jedem Servlet Code für die Anwendung bekommen. Weiterhin ist es möglich, SSL für die Datenübertragung zwischen Webbrowser und Anwendungsserver zu nutzen. Authentifizierungen sind möglich und somit auch das Schützen vor JavaScript Attacken. Die JavaEE-Sicherung schützt nicht vor "Cross-Site Request Forging" oder XSRF-Attacken.

1.8 Verfahrensentscheidung

Da der Schwerpunkt der Logik bei RAP auf der Serverseite ausgeführt wird und so keine Installation auf der Clientseite erforderlich ist, hat sich der Autor beim Vergleich der beiden Frameworks für RAP entschieden. Somit können die Vorteile von Java, wie das nachträgliche dynamische Laden von Klassen oder die Speicherbereinigung, genutzt werden.

2 Entwurf der Anwendung

Da die Applikation später auf einem Touchpanel ausgeführt wird, ist darauf Obacht zu geben, dass die Anwendung mittels Finger oder Stift einfach zu bedienen ist. So sind zum Beispiel große Buttons, zweckmäßige Abstände der Buttons und Pfeile zum Scrollen für eine einfache Bedienung durch den Anwender vorzusehen. Auch bei der Anordnung der Buttons ist es wichtig, dass diese für die Touchanwendung geeignet sind, zum Beispiel unten und rechts am Rand, damit der Bediener nicht über den ganzen Bildschirm fassen muss.

Die Anwendung soll die Maschinenzustände und Fertigungsprozessdaten visualisieren. Dabei muss es möglich sein, die Anzeige nach bestimmten Kriterien, wie nach den einzelnen Modulen oder Fehlern, zu filtern. In Abbildung 3-1 sind die möglichen Fehlermodi zu sehen. Mit jeder Betätigung des Touchfeldobjektes wechselt der Filter in die nächste Filteroption und passt dazu die entsprechende Ansicht an. Ist das erste Symbol von links auf dem Button zu sehen, wird nach dem Status „Automatik“ gefiltert, beim zweiten nach „Hand“, beim dritten nach „Stopp“ und beim letzten ist der Filter wieder deaktiviert.

Abbildung 2-1: Filtersymbole

Bei der laufenden Anzeige der Nachrichten sollen immer die letzten fünfzig einsehbar sein. Dabei werden die einzelnen Zustände entsprechend ihrer Kategorie eingefärbt. Mit dem Button, der mit dem Blitz gekennzeichnet ist, soll nach Fehlern, wie Störungen und Warnungen, gefiltert werden. Die Buttons mit dem Pfeil nach oben und unten sind für das Scrollen, wechseln der Ansicht der Tabellendaten, nach oben und unten gedacht. Die angezeigten Daten sollen zusätzlich in einer Datenbank archiviert werden, damit eine nachträgliche Auswertung erfolgen kann. Mit diesen Daten lassen sich beispielsweise Anlagenteile mit besonders hoher Ausfallrate erkennen. Über den Zeitstempel lässt sich messen, wieviel Zeit der Anlagenbediener vom Auftreten eines Fehlers (z. B. Materialmangel), bis zum Beheben der Störung benötigt. Das ist betriebswirtschaftlich besonders interessant, denn dieses Ergebnis trägt dazu bei, eine optimale Anzahl von Maschinenbedienern zu ermitteln. Weiterhin soll das aktuelle Datum mit Uhrzeit für

den Bediener immer ersichtlich sein. Am unteren Rand der Anwendung soll es die Möglichkeit geben, auf eine zweite Seite zu wechseln. Das Umschalten der Seite erfolgt über die orange eingefärbten Pfeile. Dabei wird der Pfeil der geöffneten Seite dunkelgrau hinterlegt. In der Abbildung 2-2 ist ein grafischer Entwurf dieser Ansicht zu sehen.

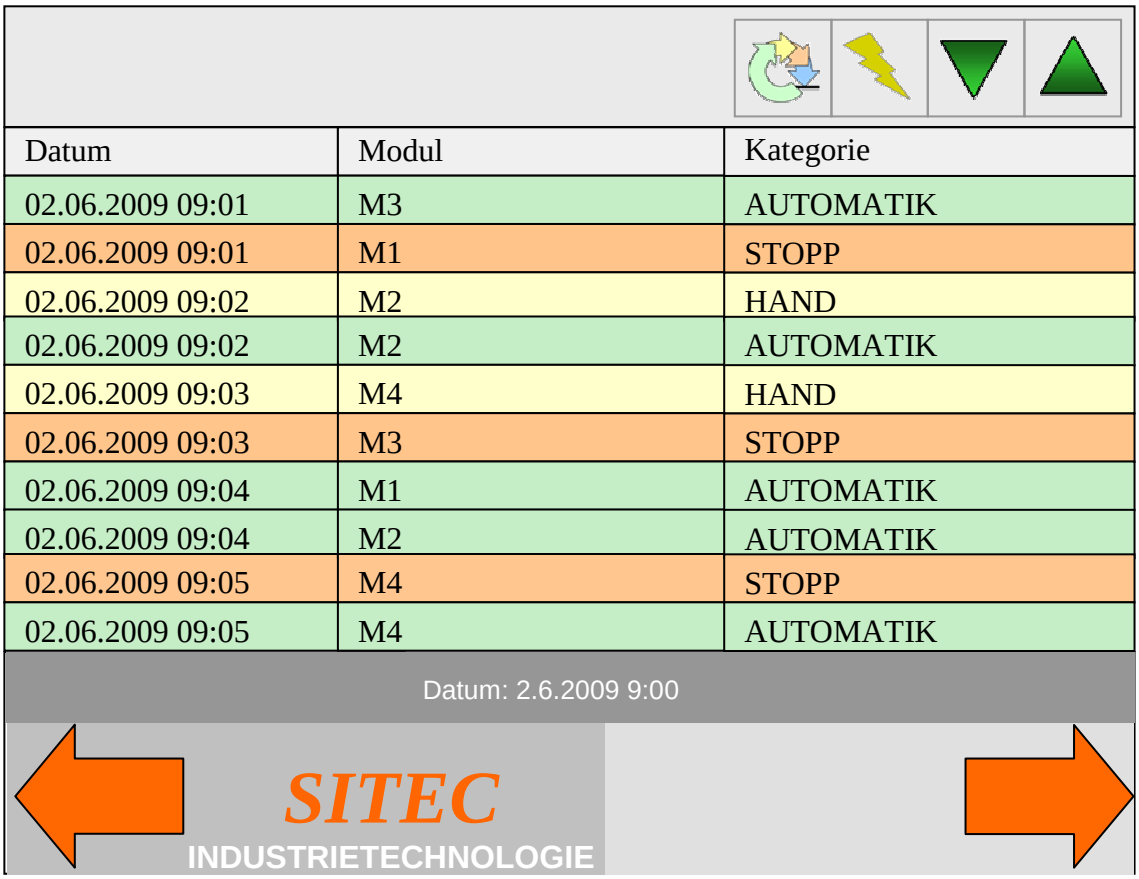


Abbildung 2-2: Entwurf der ersten Ansicht

In der zweiten Ansicht sollen immer mindestens die drei letzten Zustände zu sehen sein. Weiterhin ist eine Montageanlage mit sechs angeordneten Modulen abgebildet. Sobald ein Modul den Zustand STOPP sendet, soll dieses rot hinterlegt werden, damit der Bediener gleich sieht an welcher Stelle die Produktion gestoppt wurde. Wenn dieses Modul dann den Zustand Automatik sendet, wird die Rotfärbung wieder aufgehoben. Zudem soll auf dieser Ansicht noch die aktuelle Anzahl der IO-Produktteile, hier grün dargestellt, ausgegeben werden. Diese sagt aus, wieviele erfolgreich produzierte Teile die Anlage bereits verlassen haben. Zusätzlich soll, wie in der ersten Ansicht, das aktuelle Datum mit Uhrzeit angezeigt werden. Die Abbildung 2-3 zeigt den Entwurf der zweiten Ansicht.

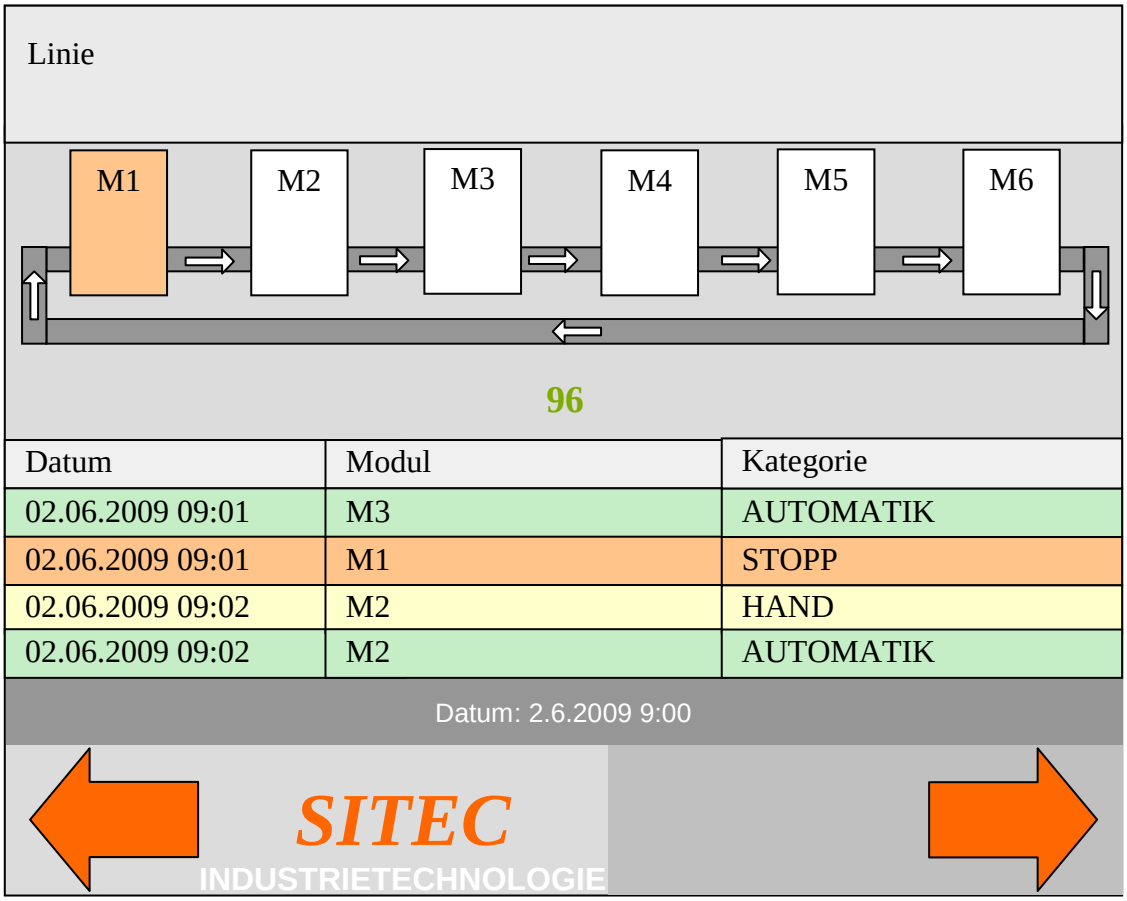


Abbildung 2-3: Entwurf der zweiten Ansicht

3 Umsetzung des Prototypen

3.1 Tabellenverwaltung

Zur Erstellung der Tabelle wurde der TableViewer gewählt, weil damit die Verwaltung der Zellen einer Tabelle unterstützt wird. Zuerst erfolgt das Setzen des LabelProviders und das Anlegen eines TableColumn-Objektes für jede Spalte. Das TableColumn-Objekt ist hier für die Erzeugung und Verwaltung der SWT TableColumn zuständig. In der Abbildung 3-1 werden die Zusammenhänge noch mal veranschaulicht.

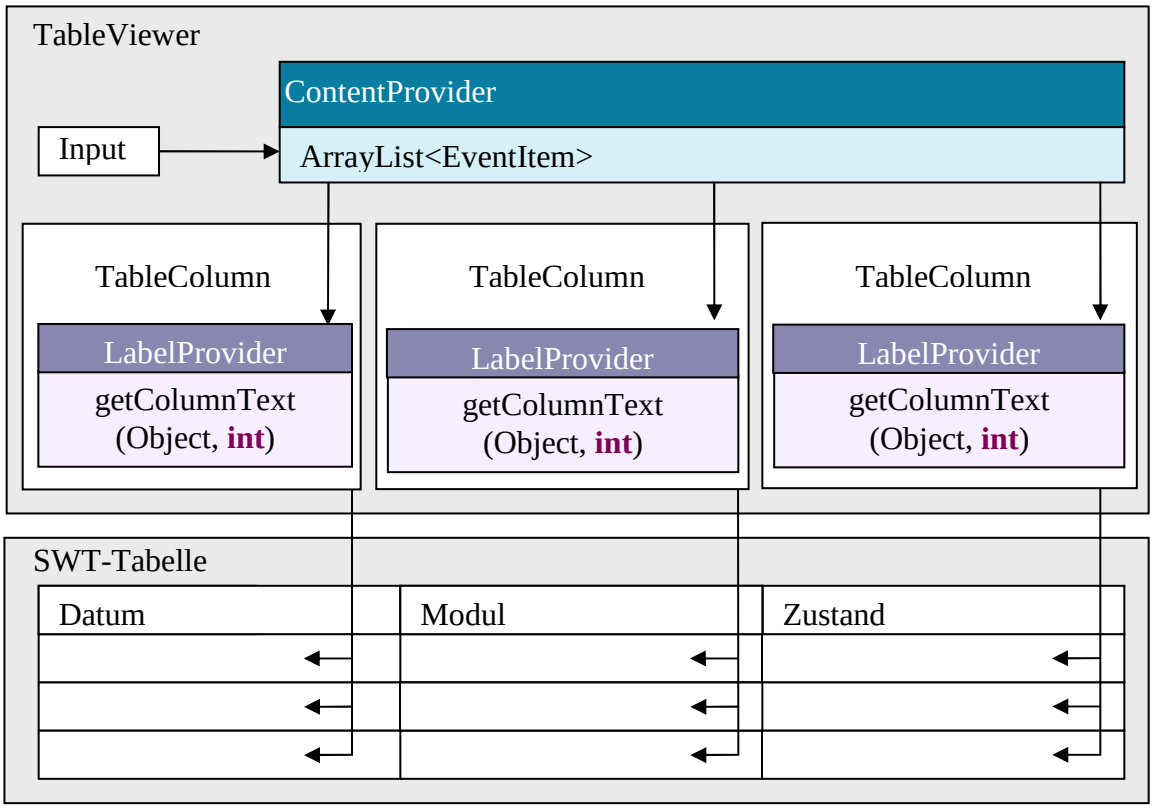


Abbildung 3-1: Aufbau eines TableViewer[7]

Nach dem Erstellen der Tabelle wurden die Spaltenbetitelung und Spaltenbreiten-festsetzung vorgenommen. Abbildung 3-2 zeigt die Festsetzung der ersten Spalte.

[7] Quelle: Eigne Darstellung

```
TableColumn column = new TableColumn(table, SWT.NONE);
column.setText("Datum");
column.setWidth(423);
column.setMoveable(true);
```

Abbildung 3-2: Erstellung einer Spalte der Tabelle

Als ContentProvider wird der ArrayContentProvider verwendet, da die TableItems als Input aus einer ArrayListe kommen. Bei den betreffenden Spalten werden dann die LabelProvider gesetzt, welche für die Befüllung der Spaltenzellen, durch einmaligen Aufruf der Zelle, zuständig sind. Und zum Schluss werden die Input-Objekte gesetzt. In Abbildung 3-3 wird die Umsetzung von ContentProviver und LabelProvider des TableViewers dieser Applikation präsentiert.

```
viewer = new TableViewer(table);
viewer.setContentProvider(new ArrayContentProvider());
        ArrayList<EventItem> events = new ArrayList<EventItem>();;
            final EventItem eventItem = new EventItem();

            eventItem.date = new Date(textMessage.getLong("date"));
            eventItem.module = textMessage.getString("module");
            eventItem.category = textMessage.getString("category");
            events.add(eventItem);

            viewer.add(eventItem);
viewer.setLabelProvider(new EventItemLabelProvider());
        public String getColumnText(Object element, int columnIndex) {
                EventItem eventItem = (EventItem) element;
                switch (columnIndex) {
                case 0:         return df.format(eventItem.date);
                case 1:         return eventItem.module;
                case 2:         return eventItem.category;
                }               return null;
        }
viewer.setInput(events);
```

Abbildung 3-3: Umsetzung des TableViewer

Das Filtern nach den Inhalten in der Tabelle wurde realisiert, indem die Methode setFilters zum Filtern der Tabelle verwendet wird. Diese Methode erzeugt eine Liste von Filter-Objekten. So werden nur die Zeilen angezeigt, bei denen kein Filter die Anzeige abgelehnt hat. In Abbildung 3-4 ist die Filterfunktion für den Zustand Stopp eines Moduls zu sehen.

```
ViewerFilter stoppFilter = new ViewerFilter() {
    public boolean select(Viewer viewer, Object parentElement, Object element) {
            EventItem eventItem = (EventItem) element;
            return "STOPP".equals(eventItem.category);
    }
};
viewer.setFilters(new ViewerFilter[] { stoppFilter });
```

Abbildung 3-4: Filterfunktion für „STOPP“

Um die Zustands-, Warnungs- und Fehlermeldungen der einzelnen Module zu kennzeichnen, wurde eine Methode getBackround erstellt. Der Methode werden die einzelnen Elemente mit ihrem Index übergeben und durch einen Vergleich des Zustandes entsprechend eingefärbt. Die Abbildung 3-5 zeig dieses Vorgehen des Markierens.

```
public Color getBackground(Object element, int columnIndex) {
            EventItem eventItem = (EventItem) element;
            if ("HAND".equals(eventItem.category)) {
                    return BG_COLOR_YELLOW;
            } else if ("AUTOMATIK".equals(eventItem.category)) {
                    return BG_COLOR_GREEN;
            } else if ("STOPP".equals(eventItem.category)) {
                    return BG_COLOR_RED;
            } else {
                    return Graphics.getColor(255, 55, 55);
            }
    }
```

Abbildung 3-5: Farbige Markierung der Objekte

3.2 JavaMessageService

Im späteren Einsatz empfängt die Anwendung die anzuzeigenden Daten von einer SPS. Um eine realitätsnahe Funktionsprüfung, ohne arbeitende Anlage und SPS-Kommunikation durchführen zu können, wird JavaMessageService (JMS) verwendet. Dieser Service steht in JavaEE bereit, um eine asynchrone Kommunikation zu realisieren. Außerdem wird der Zugriff auf nachrichtenbasierte Infrastrukturen ermöglicht. Die Kommunikation findet hierbei zwischen zwei oder mehr Clients statt. Dabei stehen zwei Arten zur Verfügung. Zum Einen kann die Kommunikation nach dem Publish-Subscribe-Prinzip erfolgen, wobei von einem Sender an mehrere Empfänger Nachrichten geschickt werden. Andererseits können Nachrichten zwischen einem Sender und einem Empfänger übertragen werden. Diese Kommunikation wird als Point-to-Point bezeichnet. Die Point-to-Point-Kommunikation erfolgt über Warteschlangen. Die Nachricht eines Senders ist genau für einen Empfänger bestimmt, dabei gibt es keine zeitlichen Abhängigkeiten. Sobald sich der Empfänger mit der Warteschlage verbindet, werden die Nachrichten vom Sender in eine Warteschlage gestellt und vom Empfänger ausgelesen. Nachdem der Empfang bestätigt wurde, wird die Nachricht aus der Warteschlange entfernt. Die Abbildung 3-6 verdeutlicht dies.

Abbildung 3-6: Point-to-Point-Kommunikation[8]

Für das Testen der hier beschriebenen Anwendung ist das Publish-Subscribe-Prinzip geeignet, da hier mehrere Empfänger möglich sind. Bei JMS erfolgt die Kommunikation über Topics. Hierbei sendet ein Client an ein Topic Nachrichten, die dann die jeweiligen Empfänger erhalten können. Dabei werden die Nachrichten erst ab dem Zeitpunkt der Clientanmeldung am jeweiligen Topic empfangen. Das heißt, dass die Nachrichten die vor dem Anmelden eines Clients an das Topic gesendet wurden, nicht mehr empfangen werden können. In der Abbildung 3-7 wird das Publish-Subscribe-Prinzip noch veranschaulicht.

[8] Quelle: Eigne Darstellung

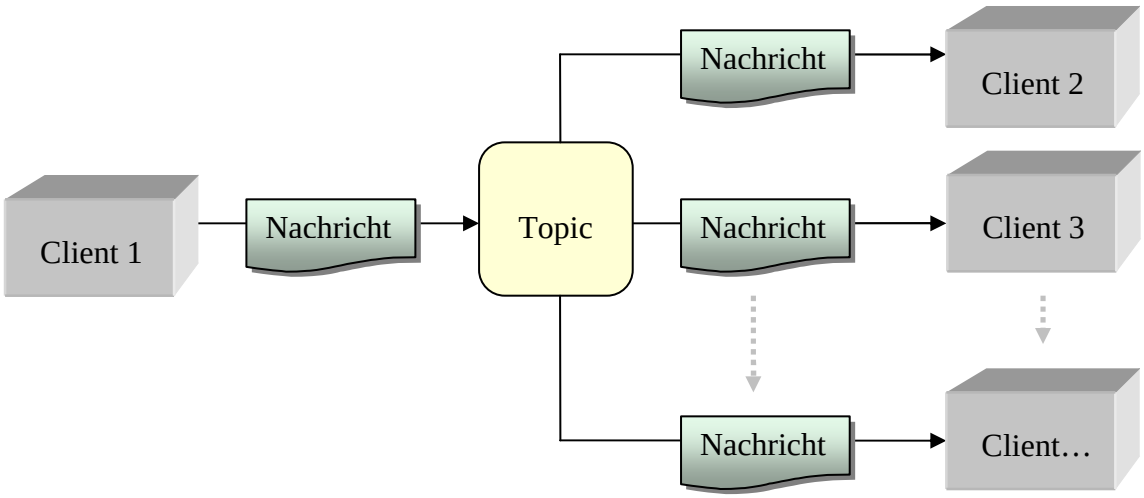


Abbildung 3-7: Publish-Subscribe-Prinzip[9]

Bei JMS gibt es verschiedene Klassen für unterschiedliche Nachrichtentypen. Der Autor verwendet MapMessage, welches zum Senden einer Reihe von Namen-Wert Paare genutzt wird. Hierbei sind die Namen String-Objekte und die Werte sind einfache Datentypen. Die Einträge können durch Namen zugeordnet werden. Beim Empfang der MapMessage durch den Client ist diese immer nur lesbar. Abbildung 3-8 zeigt das Erstellen und Befüllen einer MapMessage.

```
MapMessage message = session.createMapMessage();

message.setLong("date", datum.getTime());
message.setString("module", antworten[zufallsgenerator.nextInt(antworten.length)]);
message.setString("category", category);
```

Abbildung 3-8: MapMessage erstellen und befüllen

Als Nachrichtensystem wird Apache ActiveMQ genutzt, das ist ein OpenSource MessageBroker, welcher vollständig den JavaMessageService 1.1 implementiert. ActiveMQ fragt die betreffenden Netzwerkports fortwährend ab. In der Standard-konfiguration ist das beispielsweise "tcp://localhost:61616". Damit jetzt eine Daten-übertragung erfolgen kann, muss eine Verbindung zwischen den Clients aufgebaut werden. Bei JMS wird diese als „Connection“ bezeichnet und durch eine „Connection

[9] Quelle: Eigne Darstellung

Factory“ erzeugt. Außerdem muss eindeutig ein Endpunkt definiert werden. Dies geschieht durch die „Destination“ Festlegung und erfolgt unter Einbeziehung einer zuvor erstellten Session. Die Session erstellt weiterhin einen MessageProducer und einen MessageConsumer. In der Abbildung 3-9 wird dieser Sachverhalt noch einmal dargestellt.

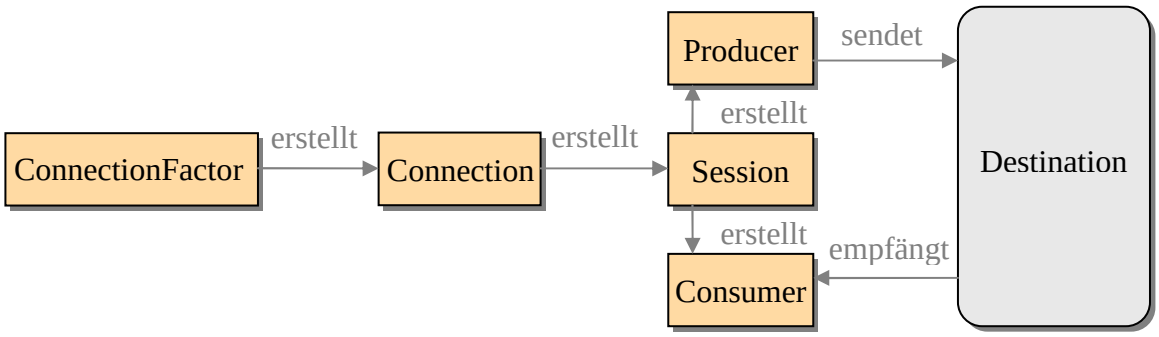


Abbildung 3-9: Aufbau einer Verbindung zwischen Clients[10]

Im MessageProducer wird also die MapMessage erstellt, mit Informationen gefüllt und abgeschickt. Damit jetzt der MessageConsumer die Nachricht empfangen kann, muss beim MessagageListener die onMessage(Message arg0) Methode implementiert werden. Dieses Argument übergibt die JMS-Nachricht und wird in der Methode den EventItems der Tabelle zugeordnet und dem TableViewer übergeben. Ein Abbild der Anwendung ist mit bereits empfangenen Nachrichten im Anhang 2 zu sehen.

3.3 Datenbank

Die abgefragten Daten werden in einer Datenbank abgelegt, damit später Fehler zurückverfolgt werden können. Als Datenbank wurde die HSQLDB gewählt, welche eine relationale SQL-Datenbank ist. Diese Datenbank wurde gezielt ausgewählt, da sie gänzlich in Java programmiert und Open-Source ist. Bei HSQL ist ein Server integriert, welcher SQL-Anfragen aus dem Netzwerk mittels JDBC entgegennimmt. Als Tabellentyp wurde sich hier für eine Logdatei entschieden, dabei werden alle SQL-Befehle, die die Daten einer Tabelle verändern, in eine SQL-Logdatei geschrieben. Solche SQL-Befehle sind CREATE, ALTER, INSERT, UPDATE. Bei jedem Start wird dann die Logdatei wieder abgearbeitet und danach gespeichert.

[10] Quelle: Eigne Darstellung

Für das Erstellen der Datenbank „eventdb“ wurde eine Stapelverarbeitungsdatei „startdb“ geschrieben, diese ist in Abbildung 3-10 abgebildet. Nachdem die Datenbank erstellt wurde und eine Verbindung zur HSQLDB Server besteht, wurde noch ein Nutzer „SA“ ohne Passwort mit dem Befehl „CREATE USER SA PASSWORD "";“ festgelegt. Der nächste Schritt war das Anlegen der Tabelle.

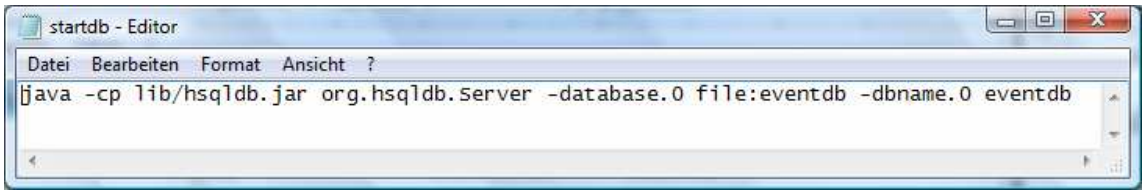


Abbildung 3-10: Die Stapelverarbeitungsdatei - startdb

Der Aufbau der Tabelle beinhaltet die Spalten Datum, Modul und Kategorie. Das Datum ist im TIMESTAMP Format. Das Modul und die Kategorie ist vom Type VARCHAR. Die Tabelle wurde dann mit dem Befehl: “CREATE MEMORY TABLE EVENTS (DATUM TIMESTAMP, MODUL VARCHAR, KATEGORIE VARCHAR);“ erzeugt.

Nachdem die Datenbank vorbereitet wurde, kann sie mit Daten gefüllt werden. Dazu werden an geeigneter Stelle im Java-Quelltext die einzelnen Nachrichtendaten den entsprechenden Variablen zugewiesen und einer weiteren Klasse zum Bearbeiten übergeben. Den entsprechenden Quellcode dazu zeigt Abbildung 3-11.

```
Long date = message.getLong("date");
String module = message.getString("module");
String category = message.getString("category");
new Daten(date, error, category);
```

Abbildung 3-11: Übergabe von Variablen zur Datenspeicherung

Die Datenspeicherung erfolgt, indem zu Beginn eine Verbindung zur Datenbank aufgebaut wird. Dazu ist es notwendig, die URL anzugeben, die anzeigt, wo sich die Datenbank befindet. Zudem muss der DB-Driver festgelegt und der Nutzernahme mit Passwort übergeben werden. Dieser Verbindungsaufbau wird in Abbildung 3-12 dargestellt.

```
String url = "jdbc:hsqldb:hsql://localhost/eventdb";
String driverName = "org.hsqldb.jdbcDriver";
String userName = "SA";
String password = "";

Class.forName(driverName);
for (Enumeration<Driver> e = DriverManager.getDrivers(); e.hasMoreElements();)
        System.out.println(e.nextElement().getClass().getName());
        try {
                Connection dbConnection =
                        DriverManager.getConnection(url, userName, password);
                System.out.println("Verbindung hergestellt");

          } catch (SQLException e) {
                System.out.println("Verbindung fehlgeschlagen");
          }
```

Abbildung 3-12: Aufbau einer DB-Verbindung

Um die Tabelle mit Daten zu füllen, wird ein Prepared Statement verwendet, welches das Datenbanksytem auf die INSERT-Anweisung vorbereitet. Hierbei werden Platzhalter übergeben, nicht Parameterwerte, wie es bei anderen Statements üblich ist. Bei unserem PreparedStatement wird erst die Richtigkeit von Parametern überprüft, bevor sie gespeichert werden. So können also in diesem Bezug keine Sicherheitslücken entstehen, die sonst beispielsweise das unerlaubte Einschleusen von Datenbankbefehlen ermöglichen. Die Befüllung der Datenbank mittels Prepared Statement wird in Abbildung 3-13 gezeigt. Und in Tabelle 3-1 ist ein Auszug der befüllten Datenbank zu sehen.

```
PreparedStatement  preparedStatement = dbConnection.prepareStatement
    ("INSERT INTO events (DATUM,MODUL,KATEGORIE) VALUES ( ?,?,? );");
preparedStatement.setTimestamp(1, new Timestamp(date));
preparedStatement.setString(2, module);
preparedStatement.setString(3, category);
preparedStatement.execute();
preparedStatement.close();

dbConnection.close();
```

Abbildung 3-13: Datenbank befüllen

DATUM	MODUL	KATEGORIE
2009-08-27 06:45:20.019	M4	AUTOMATIK
2009-08-27 06:45:22.017	M3	AUTOMATIK
2009-08-27 06:45:22.017	M3	AUTOMATIK
2009-08-27 06:45:24.014	M6	HAND
2009-08-27 06:45:24.014	M6	HAND
2009-08-27 16:35:16.416	M1	AUTOMATIK
2009-08-27 16:35:18.383	M2	HAND
2009-08-27 16:35:20.381	M4	Warnung: Füllstand Fördertopf niedrig
2009-08-27 16:35:22.379	M1	HAND
2009-08-27 16:35:24.377	M3	AUTOMATIK
2009-08-27 16:35:26.375	M4	AUTOMATIK
2009-08-27 16:35:28.372	M2	AUTOMATIK
2009-08-27 16:35:30.386	M1	STOPP
2009-08-27 16:35:32.385	M4	STOPP
2009-08-27 16:35:34.382	M3	HAND
2009-08-27 16:35:36.38	M6	HAND
2009-08-27 16:35:38.379	M4	AUTOMATIK
2009-08-27 16:35:40.377	M5	AUTOMATIK
2009-08-27 16:35:42.375	M2	AUTOMATIK
2009-08-27 16:35:44.372	M6	AUTOMATIK
2009-08-27 16:35:46.386	M1	AUTOMATIK
2009-08-27 16:35:48.384	M5	AUTOMATIK
2009-08-27 16:35:50.381	M4	STOPP
2009-08-27 16:35:52.379	M3	STOPP
2009-08-27 16:35:54.377	M6	AUTOMATIK
2009-08-27 16:35:56.375	M5	HAND
2009-08-27 16:35:58.374	M4	AUTOMATIK
2009-08-27 16:36:00.387	M5	STOPP

Tabelle 3-1: Auszug aus der Datenbank

4 Funktions- und Leistungstests

Bei den durchzuführenden Untersuchungen ist zwischen Funktionalen- und Leistungstests zu unterscheiden. Bei den Funktionstests wird getestet, ob das Programm fehlerfrei funktioniert. Einige Beispieltests sind im Anhang unter Anlage 2 zu finden. Dafür ist nur ein Nutzer notwendig. Diese Tests bilden die Grundlage für die Leistungstests. Da die Anwendung auf einem Server ausgeführt werden soll und der Zugriff über ein Netzwerk erfolgt, sind die Leistungsmessungen komplexer als bei Anwendungen, die lokal von einem Nutzer bedient werden. Zu überprüfen ist dazu die Laufzeitmessgröße, die Antwortzeit und der Durchsatz. Dabei werden die Daten im Backend durch die Anzahl sequenziell oder parallel zugreifender Nutzer interpretiert und auch die Anfragefrequenz, -komplexität und -vielfalt untersucht. Neben diesen Standardtests, zur Überprüfung der Funktionalität, und den Untersuchungen der Leistungsfähigkeit treten weitere Problematiken auf. Beispielsweise ist zu untersuchen, wie sich das System verhält, wenn mehrere Benutzer das System gleichzeitig nutzen und ob dadurch Performanceeinbußen für den einzelnen Nutzer auftreten.

Zum Verständnis werden hierfür einige Grundbegriffe erläutert und Werkzeuge zum Testen vorgestellt. Die Leistungstests werden üblicherweise als Lasttests und als Stresstests getrennt durchgeführt. Bei den Lasttests wird analysiert, wie sich das System bei hoher Anfragelast verhält bzw. welche Reaktionen treten auf. Bei Stresstests wird über die Erwartungen an das System hinaus getestet. Dabei wird beobachtet, wo und wann Probleme auftreten, ob die Antwortzeit langsamer wird und ob der Prozessor schnell genug arbeitet. Außerdem kann erkannt werden, ob zu wenig Heapspace vorhanden ist und somit ganze Anfragen verloren gehen oder ob das System dann noch antwortet. Eine Messgröße ist dabei die Antwortzeit, mit der überprüft werden kann, ob beispielsweise die Antwortzeit bei steigenden Nutzern sich erhöht.

Eine andere Größe ist der Durchsatz, mit ihm kann bestimmt werden, wieviele Transaktionen in einer vorgegebenen Zeit abschließbar sind. Hierbei werden fertige Transaktionen durch abgelaufene Zeiteinheiten dividiert. Zu beachten ist, dass eine parallele Ausnutzung der Systemressourcen höher ist als bei sequenziellen Zugriffen. Der Durchsatz verbessert sich zu Beginn bei steigender Nutzerzahl bis zu einem

Optimum, von da an sinkt der Durchsatz pro Nutzer drastisch, weil dann die Arbeitskapazität erschöpft ist.

Weitere Messgrößen sind die Anzahl der Methodenaufrufe, die durchschnittliche Rechendauer einer Methode und die Größe des Speicherplatzes, den die belegten Objekte benötigen. Zudem könnte die Größe und Auslastung des Heap-Speichers, die Einsatzhäufigkeit von Garbage Collectoren und Objekte, die noch im Speicher verblieben sind, untersucht werden.

Der Autor hat für die Untersuchung das Webserver Stress Tool angewandt. Dies ist eine Anwendung zum Testen von HTTP-Client/Server auf Leistung des Webservers. Mit dem Tool können mehrere Nutzer simuliert werden, um HTTP-Anfragen zu generieren. Des Weiteren lassen sich bei einem Leistungstest, unter Anwendung der entsprechenden URL, des Webservers oder der Webapplikation, Elemente erkennen, die eine erhöhte Ladedauer verursachen. Dadurch bietet dieser Test die Möglichkeit, Servereinstellungen zu optimieren oder durch das Testen verschiedener Realisierungen von einzelnen Webseiten/-script die Einstellungen der Applikation zu konfigurieren. Weiterhin bietet dieses Tool Ladetests, diese ermitteln die erwartete Ladedauer einer Webseite. Hierbei wird die URL angegeben, die Anzahl der Nutzer festgelegt und eine Zeit ausgewählt, die zwischen den Klicks des Webseitendatenverkehrs verstreichen soll. Zudem bietet das Webserver Stress Tool einen Stresstest. Bei diesem erfolgen mit Simulationen von Brute-Force-Attaken das exzessive Laden vom Webserver und massive Nutzeraktivitäten. Ein Stresstest bietet also die Möglichkeit, Datenverkehrsgrenzen herauszufinden. Besonders interessant ist auch ein Testverfahren, bei dem die maximal mögliche Anzahl der Nutzer über einen gegebenen Zeitraum ermittelt wird. Die höchstmögliche Anzahl ist erreicht, wenn der Webserver beginnt, Fehlernachrichten zu verursachen beziehungsweise offene Anfragen unbeantwortet bleiben.

Für den korrekten Ablauf der Tests dieser Applikation muss sichergestellt sein, dass das Netzwerk sowie die verwendeten Testrechner nicht an den Leistungsgrenzen arbeiten. Die Testergebnisse spiegeln sonst ausschließlich die Leistungsgrenzen der Rechentechnik wieder. Daher wurde der Test an einem Netzwerk mit einer angegebenen Datenübertragungsrate von 54 Mbit/s durchgeführt. Während des Testes hatte der Test-PC fortwährend ausreichende Ressourcen, so dass ein CPU-Load von zehn Prozent

nicht überschritten wurde und 1 GB freier Arbeitsspeicher stets vorhanden war. Letztendlich ist ein Softwareleistungstest natürlich von der Leistungsfähigkeit der verwendeten Rechentechnik abhängig, daher sind in Tabelle 4-1 die Rahmenbedingungen für diesen Test festgehalten.

Test Fall	Test Typ: 30 Minuten Dauertest mit RAMP Nutzersimulation: bis 10 simulierte Nutzer - 5 Sekunden zwischen den Klicks Intervall: protokolliert alle 15 Sekunden
URLs	URL Ablaufplanung: Nutzer klicken die gleiche URL an (zum verteilten gleichmäßigen Laden auf allen URLs) URL: Local IPs: http://192.168.2.110:49551/rap?startup=view
Browser Einstellungen	Mozilla/5.0 HTTP Anfrage, Zeitbeschränkung: 120 s
Client System	System Windows XP V5.1 (Build 2600) Service Pack 2, CPU Proc. Lev. 686 (Rev. 5894) at 2393 MHz, Memory 1426 MB available RAM of 2088 MB total physical RAM, 3422 MB available pagefile, 24771 MB free disk space on C:
Test Software	Webserver Stress Tool: 7.2.2.261 Trial Version

Tabelle 4-1: Einstellungen für den Test

Bei diesem Testszenario wurde über einen Zeitraum von 30 Minuten die Anzahl der Nutzer von eins auf zehn erhöht. In der Abbildung 4-1 sind die erfassten Ergebnisse eines jeden Nutzers zu sehen. Die Auswertung erfolgte bezüglich Klicks mit jeweiligen Treffern, Fehler die aufgetreten sein können, durchschnittliche Klickzeit in Millisekunden, übertragene Bytes und erreichten Kilobits pro Sekunde. In der Abbildung 4-1 wird dann dazu eine Grafik der gesendeten und empfangenen Anfragen, dem Netzwerkverkehr und den offenen Anfragen präsentiert. Die Abbildung 4-1 zeigt auf der X-Achse die steigende Nutzeranzahl, welche mit laufender Zeit schrittweise steigt. Auf der Y-Achse sind einmal in grün die gesendeten Anfragen zu sehen, welche sich annähernd deckungsgleich zu den hellblau eingefärbten empfangenen Anfragen verhalten. Hieraus ist erkennbar, dass keine Fehler aufgetreten sind. Der Netzwerkverkehr verläuft annähernd proportional zur Nutzeranzahl. Wenn

beispielsweise das doppelte Datenvolumen empfangen werden soll, wird auch die doppelte Datenübertragungsrate in Anspruch genommen. Dies geschieht solange, wie noch ausreichend Bandbreite vorhanden ist. Aus dem Diagramm ist ersichtlich, dass bei bis zu fünf Nutzern eine Anfrage offen bleibt. Das bedeutet, bis zu fünf Nutzern gibt es noch keine Leistungseinbußen. Ab sechs bis neun Nutzern treten schon zwei offenen Anfragen auf. Bei zehn Nutzern erfolgen dann bis zu zwei Klicks pro Sekunde, dies führt bis zu vier offenen Anfragen.

Nutzer	Klicks	Treffer	Fehler	Durchschnittliche Klickzeit [ms]	Bytes	Kbits/s
1	354	353	0	27	1.543.305	1.314,83
2	325	324	0	36	1.416.526	964,46
3	297	296	0	27	1.294.102	1.279,13
4	269	268	0	28	1.170.723	1.241,74
5	241	240	0	28	1.049.271	1.238,54
6	212	211	0	23	920.591	1.514,14
7	184	183	0	30	800.061	1.162,79
8	156	155	0	33	677.639	1.045,65
9	128	127	0	25	555.232	1.384,24
10	99	98	0	24	428.424	1.463,37

Tabelle 4-2: Ergebnisse pro Nutzer

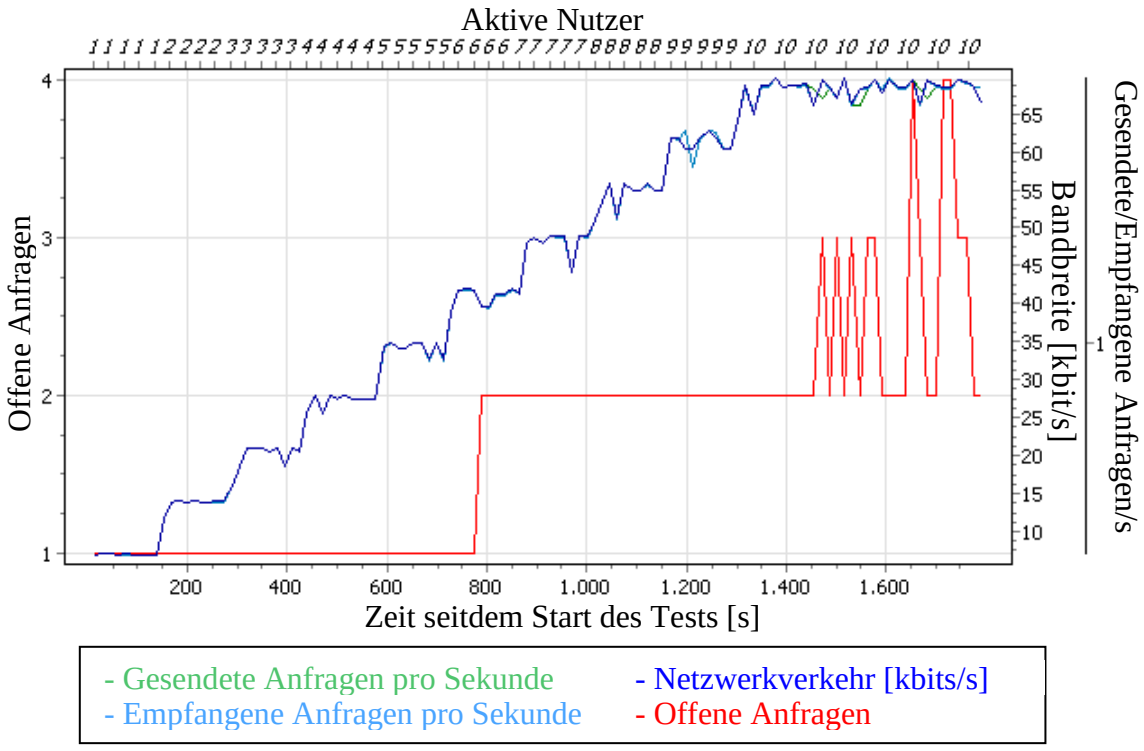


Abbildung 4-1: Offene Anfragen und übermittelte Daten

5 Zusammenfassung und Ausblick

Die in dieser Arbeit entwickelte Webanwendung ist ein Prototyp und soll eine Grundlage für den späteren Einsatz in der Montageanlagenüberwachung darstellen. Der Schichtleiter kann damit von jedem PC im Firmennetzwerk mittels Webbrowser auf die Maschinendaten einer Anlage zugreifen. Er muss dazu nicht erst eine Programm-installation durchführen und sich auch nicht an die Maschine in der Produktionshalle begeben.

Es wurden verschiedene Möglichkeiten zur Visualisierung ausgesucht und nach relevanten Kriterien analysiert. Als dann für die Anwendung eine optimale Lösung zur Umsetzung gefunden wurde, begannen die Gestaltungsrealisierungen. Bei der Erarbeitung des Layouts wurde darauf geachtet, dass dieses ergonomisch ist und so dem Anwender eine angenehme Bedienoberfläche bietet. Nachdem das Konzept für die Gestaltung der GUI stand, begann die prototypische Umsetzung der Anwendung. Dabei wurde die Software so aufgebaut, dass sie für weitere Projekte leicht wiederverwendbar ist und so dem Programmierer viel Zeit erspart.

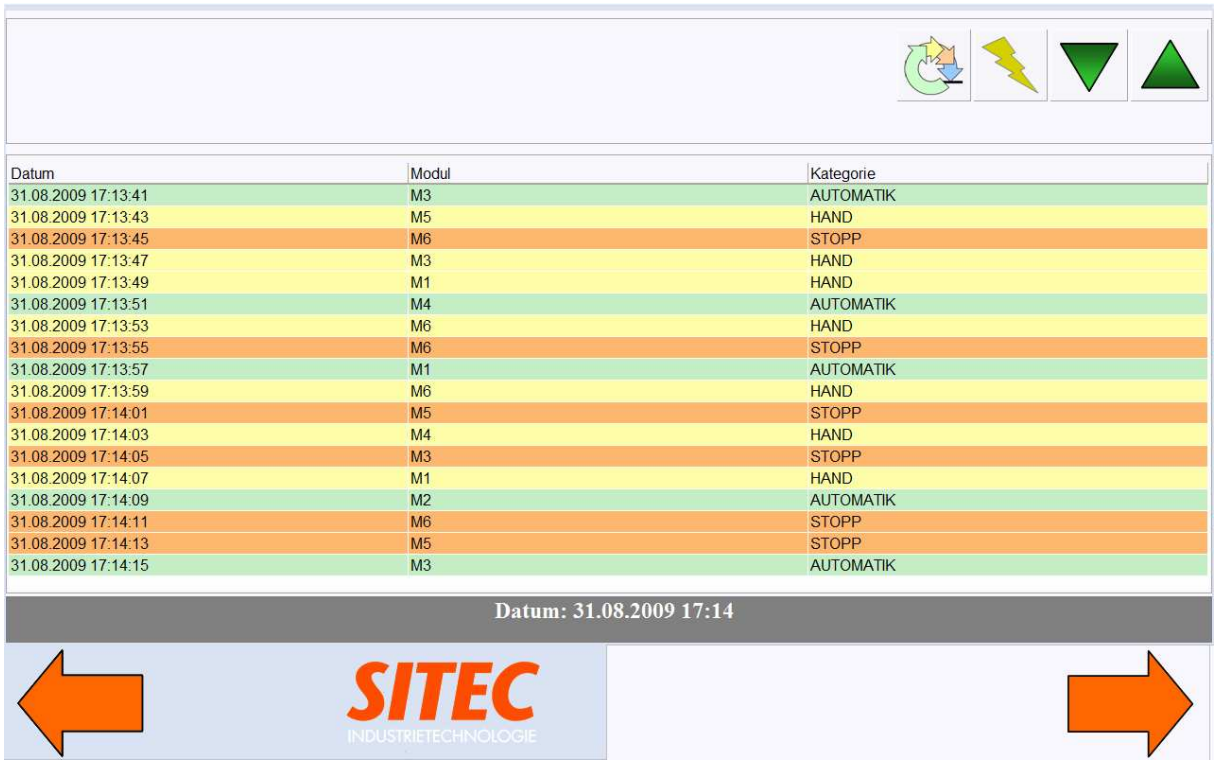
Abschließend ist zu sagen, dass das Ergebnisse der Bachelorarbeit eine gute Grundlage für die Visualisierung künftiger Sondermaschinen bietet. Ein Ausbau von Webapplikationen ist in den höheren Steuerungsebenen einer Anlage denkbar. So könnte in dieser Anwendung ein Schichtplan integriert werden, in dem die Sollstückzahl vorgegeben wird. Am Ende einer Schicht ist infolgedessen ein Soll- / Ist-Vergleich durchführbar. Der Produktionsleiter erhält so einen Überblick über die Erfüllung des Schichtplans. Es sollte zudem noch eine Nutzerverwaltung für die Anmeldung integriert werden, damit sich nur Personen mit entsprechenden Rechten auf die Anlage einwählen können. Dafür ist der Anwendung noch eine einblendbare Bildschirmtastatur hinzuzufügen, über welche dann auch Eingaben für einen Schichtplan möglich sind. Letztendlich sind in diesem Zusammenhang noch viele Erweiterungen denkbar, auch durch die ständige Weiterentwicklung von RAP werden in Zukunft neue Funktionen zur Verfügung stehen.

Anlagen

Anlage 1 - Webframeworks

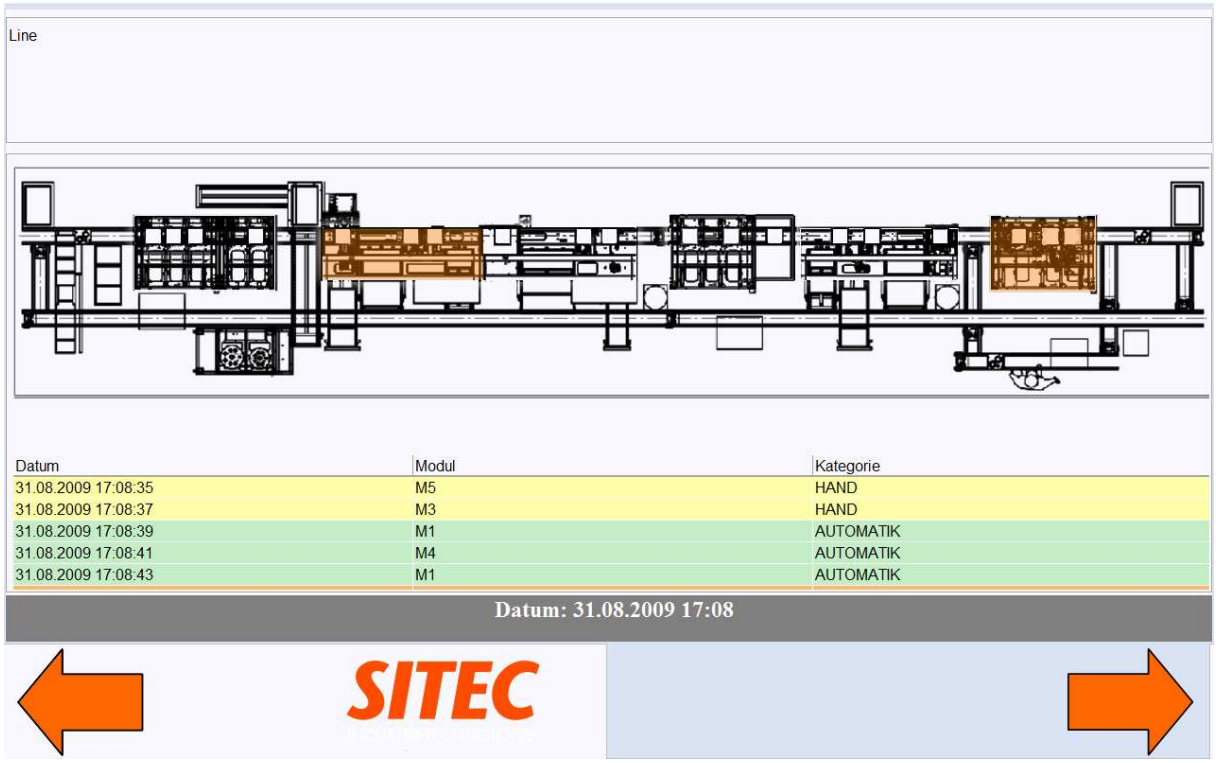
Hersteller	*Frame-work*	*Version*	*Sprache*	*Ajax*	*MVC*	*Inter-national isierung*	*Tests*	*Saubere Trennung Anwendung und Darstellung*	*Logik Schwer-punkt*	*Frame-work*
Apache	Struts 2	2.1.6	Java	Ja	Ja, Pull	Ja	JUnit	Nein, verwendet JSP Seiten	Server	Web
Google Inc.	GWT	1.6	Java	Ja	Nein, Pull	Ja	JUnit + GWT itegration, Browseremulator	Ja, Keine JSP	Client	Web+ RIA
Spring-Source	Spring	3.0.0. M3	Java + XML	Nein	Ja, Push	Ja	JUnit 4 TestNG	Nein, JSP	Server	Web
Apache	Tapestry	5.1.0.5	Java	Ja	Ja, Pull	Ja	JUnit, Abstract-IntegrationTest Suite mit Selenium	Ja	Server	Web
Apache	Wicket	1.4	Java	Ja	Ja, Pull	Ja	JUnit, Class-BaseWicketTester	Ja	Server	Web
Sun	Java-Server Faces	2.0	Java	Ja	Ja	Ja	JUnit, JSFUnit	Nein, nur unter Einsatz von Faceletts	Server	Web
Laszlo Systems	Open-Laszlo	4.3	Java Script und XML	Nein	Ja, Push	Ja	JUnit, LzUnit, Browser Emulator	Ja	Client	Web+ RIA
Spring Source	Grails	1.1.1	Java, Goovy	Ja	Ja, Push	Ja	JUnit, inegration tests, funktionstest	Ja	Server	Web
NextApp	Echo 2	2.1.0	Java	Ja	Nein, Pull	Nein	JUnit, Browser Emulator	Ja	Server	Web
Eclipse	RAP	1.2	Java	Ja	Ja, Pull	Ja	JUnit, PushToTest mit Selenium	Ja	Server	Web+ RIA

Anhang 2 – Abbild der Anwendung



Datum	Modul	Kategorie
31.08.2009 17:13:41	M3	AUTOMATIK
31.08.2009 17:13:43	M5	HAND
31.08.2009 17:13:45	M6	STOPP
31.08.2009 17:13:47	M3	HAND
31.08.2009 17:13:49	M1	HAND
31.08.2009 17:13:51	M4	AUTOMATIK
31.08.2009 17:13:53	M6	HAND
31.08.2009 17:13:55	M6	STOPP
31.08.2009 17:13:57	M1	AUTOMATIK
31.08.2009 17:13:59	M6	HAND
31.08.2009 17:14:01	M5	STOPP
31.08.2009 17:14:03	M4	HAND
31.08.2009 17:14:05	M3	STOPP
31.08.2009 17:14:07	M1	HAND
31.08.2009 17:14:09	M2	AUTOMATIK
31.08.2009 17:14:11	M6	STOPP
31.08.2009 17:14:13	M5	STOPP
31.08.2009 17:14:15	M3	AUTOMATIK

Ansicht 1 - ungefiltert



Datum	Modul	Kategorie
31.08.2009 17:08:35	M5	HAND
31.08.2009 17:08:37	M3	HAND
31.08.2009 17:08:39	M1	AUTOMATIK
31.08.2009 17:08:41	M4	AUTOMATIK
31.08.2009 17:08:43	M1	AUTOMATIK

Ansicht 2 – Modul 2 und 6 sind im Zustand STOPP

Anlage 3 – Funktionstest

TESTSZENARIO	FUNKTION	BEDINGUNG	SOLL	IST
filtern nach dem Zustand „Automatik“	klicken auf grünen Filterbutton	Nachrichten mit Zustand „Automatik“ vorhanden	nur Module mit Zustand „Automatik“ anzeigen	zeigt nur Module im Zustand „Automatik“ an
filtern nach Warnungen und Störungen	klicken auf Symbol „Blitz“	Warnungen oder Störungen müssen eingetreten sein	nur Warnungen oder Störungen anzeigen	es werden nur Warnungen angezeigt
scrollen der Tabelle nach oben	klicken auf Symbol „Pfeil nach oben“	Tabelle muss ausreichend gefüllt sein	Ansicht soll sich nach oben verschieben	Ansicht verschiebt sich nach oben
wechseln in Ansicht 2	klicken auf Symbol „Pfeil nach rechts“	Ansicht 1 ist angewählt	2. Ansicht soll sich öffnen	Ansicht 2 ist offen
markieren des gestoppten Moduls	-	Eintritt des Zustandes Stopp eines Moduls	das entsprechende Modul sollte sich rot verfärben	Modul 1 färbt sich rot
prüfen des DB Eintrages	-	Daten wurden empfangen	Datensätze sollen in der DB stehen	Datensätze stehen in der DB
wechseln in Ansicht 1	klicken auf Symbol „Pfeil nach links“	Ansicht 2 ist angewählt	1. Ansicht soll sich öffnen	1. Ansicht ist offen

Literaturverzeichnis

Apache: Tapestry. URL: <http://tapestry.apache.org/>. (Abruf am: 12.06.2009)

Apache: Wicket. URL: < http://wicket.apache.org/>. (Abruf am: 18.06.2009)

Eclipse: Rich Ajax Platform (RAP). URL: <http://www.eclipse.org/rap/>.
(Abruf am: 12.06.2009)

Google: Google Web Toolkit. URL: <http://code.google.com/intl/de-DE/webtoolkit/>.
(Abruf am: 26.06.2009)

HSQL database engine: HSQL. URL: <http://www.hsqldb.org/web/
hsqlDocsFrame.html> .(Abruf am: 05.08.2009)

Laszlo Systems: OpenLaszlo. URL: <http://www.openlaszlo.org/>.
(Abruf am: 23.06.2009):

Nextapp: Echo2. URL:< http://echo.nextapp.com/site/echo2>. (Abruf am: 17.06.2009)

Paessler: Webserver Stress Tool. URL: http://download-cdn.paessler.com/download/
webstressflyer.pdf(Abruf am: 20.08.2009)

Rudlof, Christiane: Handbuch der Software-Ergonomie. URL:< http://www.ukpt.de/
pages/dateien/software-ergonomie.pdf>. (Abruf am: 24.06.2009)

spring source: Grails. URL:< http://grails.org/>. (Abruf am: 16.06.2009)

spring source: Spring. URL: <http://www.springsource.org/about>.
(Abruf am: 30.06.2009)

Sun: JavaServer Faces. URL: <http://java.sun.com/javaee/javaserverfaces/>.
(Abruf am: 25.06.2009)

The Apache Software Foundation: Active MQ. URL:<http://activemq.apache.org/>. (Abruf am: 16.07.2009)

The Apache Software Foundation: Struts 2. URL:< http://struts.apache.org/2.x/index.html>. (Abruf am: 10.06.2009)

Wulz, Dieter: Anwendung unter Last, in: Javamagazin, Februar 2008, 3.08:S.15-18.

Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Chemnitz, 03.09.2009 — Bearbeitungsort, Datum

Kathleen Fiegert — Unterschrift